Type Uncertainty in Ontologically-Grounded Qualitative Probabilistic Matching

David Poole[1] and Clinton Smyth[2]

[1] Department of Computer Science, University of British Columbia
[2] GeoReference Online Ltd., http://www.georeferenceonline.com

Abstract. This paper is part of a project to match real-world descriptions of instances of objects to models of objects. We use a rich ontology to describe instances and models at multiple levels of detail and multiple levels of abstraction. The models are described using qualitative probabilities. This paper is about the problem of type uncertainty; what if we have a qualitative distribution over the types. For example allowing a model to specify that a meeting is always scheduled in a building, usually in a civic building, and never a shopping mall can help an agent find a meeting even if it is unsure about the address.

1 Introduction

In a recent paper [Smyth and Poole, 2004], we described a system for matching instances and models of real-world phenomena. These instances and models have been described by different people using controlled vocabularies (using an ontology) which allow descriptions of model and instances at varied levels of abstraction (using more general or less general terms) and detail (describing objects in terms of parts and subparts or not).

In one practical domain, geological surveys of various countries or provinces publish descriptions of mineral occurrences[3] (e.g., twelve thousand in British Columbia) at widely varying levels of abstraction and detail. Other people spend careers developing models of archetypical mineral occurrences that can help determine where certain minerals can be found in sufficient quantities to be mined. These models are often at different levels of abstraction and detail from the occurrence descriptions. The problem that we consider is to determine which mineral occurrences fit which models, so that explorers can focus their exploration. This is a case where humans have to make decisions, but they are overwhelmed by the combinatorics. The aim of the computer system is to find the best fitting models to a mineral occurrence or to find the best fitting mineral occurrences to a model. As it is humans who are making the decisions, it is more important to have good explanation facilities and to return multiple potential matches than to return the "best" match according to the computer. Computers help by narrowing down the search space and explaining and justifying the potential matches.

[3] In nature there exists a continuum between mineral occurrences and mineral deposits. Geologists tend to call small accumulations of minerals "mineral occurrences", and large ones "deposits". It may take many years to determine whether a mineral occurrence is large enough to be thought of as a deposit.

The main problem is the integration of (qualitative) probabilistic reasoning and the rich ontologies that are needed in such domains. For this paper we'll use the OWL [McGuinness and van Harmelen, 2004; Patel-Schneider, Hayes and Horrocks, 2004] notation where appropriate.

In previous work [Smyth and Poole, 2004], we make the assumption that different descriptions refer to different objects. This assumption is relaxed in this paper. In particular, we allow for uncertainty in the types and allow for qualitative distributions over hierarchically structured classes. By type we mean membership in a class, where the classes are organised hierarchically and are specified as part of an ontology.

This work is quite different to other work on combining probability and ontologies (e.g., P-Classic [Koller, Levy and Pfeffer, 1997]) because we are using the ontologies to construct a rich hypothesis space rather than only having probabilities over the ontologies.

2 Qualitative Probabilistic Matching

The general problem is, given a model and a instance, to determine a qualitative value for $P(model|instance)$ that can be used by a human to make decisions. This section gives an overview of our previous paper [Smyth and Poole, 2004].

We decided to use qualitative order-of-magnitude probabilities based on the kappa calculus [Spohn, 1988; Pearl, 1989]. The kappa calculus can be described in terms of surprise; the kappa values correspond to the level of surprise. When probabilities multiply, the corresponding kappa values add, and summing in probability corresponds to minimization in the kappa calculus.

The kappa calculus can be seen as a crude approximation to probability [Darwiche and Goldszmidt, 1994]. This is useful when the probabilities are not available (and may be different for different users) as it gives a rough answer and leaves only a few of the possible matches for a human to evaluate; the implausible matches can be ignored by the human. Note that we use the kappa calculus for the first-level of approximation; we use some finer distinctions to distinguish matches that may have the same values in the kappa calculus. These are described when used.

The instances will have property values that are marked as "present" or "absent".

After feedback from domain experts, we describe the models using a 5 value scale[4]:

- *always p*: you are very surprised if p is false[5].
- *usually p*: you are somewhat surprised if p is false.
- *sometimes p*: you aren't surprised if p is true or if it's false
- *rarely p*: you are somewhat surprised if p is true.
- *never p*: you are very surprised if p is true.

[4] None of the theory or results in this paper depends on using this scale, but we will use it in all of our examples. In practice, we have found that experts are happy using this scale, and find it very natural.

[5] This is *not* the always of modal logic. Our experts described things as "always" true even though there were exceptions.

These provide a language for inputting our measures of uncertainty. We output a numerical value in the range [0,100] where 100 is the score for the best possible match and 0 is the score for the worst match. Internally we use a reasonably arbitrary numerical scale that we will use in this paper.

If not for different levels of abstraction and different levels of detail, to compute the qualitative counterpart of $P(model|instance)$, we add the *surprises* of the instance with respect to the qualitative probabilities specified in the model. The main contribution of [Smyth and Poole, 2004] was in showing how the kappa calculus could be combined with rich ontologies that let us describe models and instances at various levels of abstraction and detail.

As we are adding surprises, and returning the topmost (i.e., least surprising) match, the zero point is arbitrary. We can define the zero to be the level of the empty match[6] (i.e., a match with an empty description), then we have positive rewards when there is a better match than this and negative rewards (penalties) for those matches that are worse than this.

For each model qualitative probability and for each instance value “present” or “absent”, we will have a numerical reward or penalty. Thus, for example, we will talk about the always-present reward (which gives the reward received when a model property that has qualitative probability “always” matches an corresponding instance property that is present) or a usually-absent penalty (when the model property is “usually” present, but it is absent in the instance). For example, if a model specifies a room that is always a bedroom and usually pink and we have an instance that is a bedroom that is not pink (i.e., bedroom is present and pink is absent), we get both the always-present reward and the usually-absent penalty.

Given an abstraction hierarchy of classes, it is important to distinguish the description of an instance from the instance itself. For example, if something is described as a building, it must be some sort of building (generic buildings don’t exist). One of the differences between an instance and a model is, when given a general concept, such as “building”, in an instance we *don’t know* what sort of building it is, but if the same term is used in a model, we *don’t care* what sort of building it is [Smyth and Poole, 2004]. When we want the probability of an instance, we don’t want the probability of the description. The probability of a more abstract description is more likely than that of a more specific instance. For or example, a house is a kind of building, so for any evidence e, $P(building|e) > P(house|e)$. However if the model specifies a house, and instance 1 is described as a building and instance 2 is described as a house, then instance 2 definitely fits the model, but it is less likely that instance 1 fits the model.

Smyth and Poole [2004] made two assumptions that we relax in this paper:

- The type of objects was known. That is, there was no qualitative distribution over which class an object is a member of. It did not allow, for example, the model to specify that a place that can take the role of a home office is always a room, usually a bedroom and rarely a master bedroom.

[6] This is for the case of the open-world assumption, where we don’t assume that a complete description is given. We do allow someone to specify that “silence implies absence”; that a part or property that is not described is false. In this case an empty description does not have value zero. It is positive as we expect nothing else and found nothing else.

- In a single description, different descriptions of parts (or other property values) pertain to different parts (or property values). Consider the following two contrasting examples:
  - Suppose we had a a model of "a room that Sam likes" that says the room is "usually red and never pink". This could mean that it has a single colour that is usually a non-pink shade of red. Alternatively, there could be multiple colours; as long as one is red and one is non-pink they would be happy. So a blue and pink would be good; they just don't want all-pink or no shade of red. In the first case, the description is referring to a single colour and in the second to multiple colours. Here it seems intuitive that this description is about the same colour.
  - Suppose we have a model for a house that always contains a bathroom and always contains a room that is not a bathroom. In this case, we don't want this to refer to the same room (there is no room that is both a bathroom and is not a bathroom), but rather to two rooms.

  We need a way to specify we are referring to a single colour or room or to multiple colours or rooms.

Note that these assumptions are interdependent; if we relax the second assumption, we could relax the first by treating class membership as a functional property. However, it is easier to motivate the problems in terms of types, but then treat type as a (functional) property in the algorithm.

3 Type Uncertainty

We assume that we have a hierarchy of classes and a hierarchy of properties.

Figure 1 shows a hierarchy where, some of the relations that are true include:

$$\langle House, subClassOf, ResidentialBuilding\rangle$$
$$\langle ResidentialBuilding, subClassOf, Building\rangle$$
$$\langle Building, subClassOf, EngineeredArtifact\rangle$$

For this discussion, we do not intend that these are immediate subclasses. There may be many intermediate classes (e.g., that are classes that are subclasses of "Engineered Artifact" and superclasses of "Building"). In this figure, *Thing* is the topmost class.

Suppose a model that specifies that the location of some activity is:

always a *Building* and *rarely* a *House*.

Call this model M_1.

Given a simple description of an instance where we only give a single class that is present, let's determine how surprised we are that the location is at that instance. The description could be any position in the taxonomy, and the instance could be any leaf that is a descendent of the class that is said to be present. The figure shows the five qualitatively different regions of the taxonomy that the description of an instance could be in:

- Region 1. If the description is in region 1, the cousins of *Building* (the values in the same tree that are neither descendents nor ancestors of *Building*), the instance is not a *Building*.

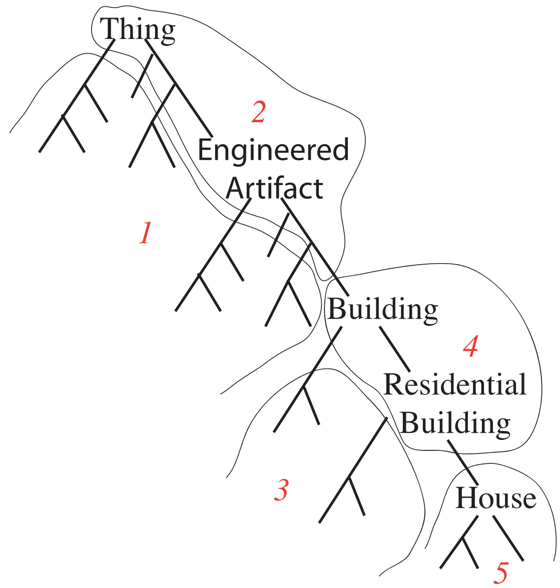

Fig. 1. A class hierarchy. Different exceptional regions for model M_1 are numbered.

- Region 2. The description is an ancestor of *Building* so the instance is perhaps a *Building*, but could be a non-building.
- Region 3. The instance is a *Building* and not a *House*.
- Region 4. The instance is a *Building* and perhaps a *House*.
- Region 5. The instance is a *House*.

Consider how surprised we would be that an object in each of these locations would be an instance of the model "*always* a *Building* and *rarely* a *House*".

If the instance description is in region 1, the instance would receive the always-absent penalty. The model says the instance is always a building and the instance is not a building.

Suppose description d_2 of the instance is in region 2, for example it is the description *EngineeredArtifact*. We don't know if the instance is a *Building* or not. To understand the qualitative probability, consider the probability of the model M_1 given the description d_2:

$$P(M_1|d_2) = P(M_1|Building \wedge d_2) * P(Building|d_2) \\ +P(M_1|\neg Building \wedge d_2) * P(\neg Building|d_2)$$

$P(M_1|Building \wedge d_2) = P(M_1|Building)$ as the model doesn't specify anything more general that *Building*. Similarly, $P(M_1|\neg Building \wedge d_2) = P(M_1|\neg Building)$. Thus

$$P(M_1|d_2) = P(M_1|Building) * P(Building|d_2) \\ +P(M_1|\neg Building) * P(\neg Building|d_2)$$

In terms of the kappa-calculus (taking logs, ignoring adding by zero, and minimizing):

$$\kappa(M_1|d_2) = min(\kappa(Building|d_2), \kappa(M_1|\neg Building)$$

assuming that

- $\kappa(\neg Building|d_2) = 0$; we are not surprised that the *Engineered Artifact* is not a building. We would be surprised if it is a building as there are many more sorts of engineered artifacts than buildings.
- $\kappa(M_1|Building) = 0$; we are not surprised that a building matches the model M_1 as the model M_1 specifies the object is always a building.

Thus the "surprise" that the engineered artifact fits the model comes from either the surprise that the *Engineered Artifact* is a building or the surprise that a non-building matches the model. Our level of surprise is the minimum of these two.

We could have surprise information as part of our ontology. For each element in the taxonomy, we would have a value of how surprising each child is. For example, we could infer the surprise of *Building* given the description *Engineered Artifact*.

If we didn't have the information in the taxonomy, we can make some simplifying assumptions to estimate this value. Suppose the description d_2 is m levels in the hierarchy above *Building*, and suppose that the average branching factor of b, and that the children of any node have approximately equal probability. Then $P(Building|d_2) \approx (1/b)^m$. Taking logarithms, we see that the surprise that the instance is a *Building* should be linear in m.

We do not use the kappa calculus directly, as this would entail having a surprise that there is no description. We'd rather just ignore non-existent descriptions. This can be done by defining the surprise value of a empty description as zero. We get positive rewards for being less surprised than this and negative rewards (penalties) for being more surprised. Given that we know something exists, not specifying a value is the same as stating it has the top value (*Thing* in the above taxonomy). This then gives us a way to calibrate the surprise due to the reasons above. The value is zero when the description is *Thing*. If the description is not *Thing*, but in region 2, then is should have a positive reward, as it is more likely a *Building* than if it were a *Thing*. Under the assumptions made above, this should be linear in the depth. The main assumption is that surprises are not given in the taxonomy and that all children are approximately equal[7].

If the assumption that children are approximately equal is not a reasonable assumption, it is possible to specify the surprise values for each child in the taxonomy as part of the ontology. For example, specifying how surprised you are that residential building is a house. Note that in this case it is possible to specify a model that is surprised by a normal condition; in this case, the model should also be surprised by a empty description. For example, if things are usually engineered artifacts, but a model specifies that fits to the model are rarely engineered artifacts, then the model should be surprised by a description of an object just as *Thing*.

[7] Note that this means all children along the path from *Thing* are approximately equal, not that a child is equal to all of its sibling

If the instance is in region 5, it gets the rarely-present penalty. We know the instance is a type of house but the model specifies that we should be surprised by the meeting in a house.

In order to understand the reward of an instance in region 3, it is instructive to consider some more models. Suppose model M_0 is "*always* a *Building*" and M_2 is "*always* a *Shopping Centre*". Then we have M_2 subsumes M_1 (given that *Shopping Centre* is a subClassOf of *Building* and is disjoint with *House*) and M_1 subsumes M_0. If the instance is a *Strip Mall* (a subClass of *Shopping Centre*), then this instance matches all three models. As M_0 and M_2 give the same match; an always-present reward, it seems reasonable to also give the match with M_1 the same reward and to not also give it a rarely-absent reward.

Instances in region 4, are known to be buildings and they could be houses. If we do a similar probabilistic analysis to region 2, with d_4 a description on region 4, we get:

$$\begin{aligned} P(model|d_4) = & P(model|House \wedge Building) * P(House|d_4) \\ & + P(model|\neg House \wedge Building) * P(\neg House|d_4) \end{aligned}$$

(as $P(Building|d_4) = 1$). If you just consider the second part of the sum, you are not surprised that the model is true for a building that is not a house (it is "always" true), you are also not surprised that a building in region 4 is not a house. Thus in terms of the kappa values, this has kappa value 0. That is, $kappa(model|d_4) = 0$.

However, this is considered to be a worse match than for an instance in region 3, and so gets a small penalty that is proportional to the depth of the description. This value is designed to be dominated by the kappa values so that it only distinguished instances that have the same kappa values.

4 Matching Algorithm

For this paper[8], we assume that we are given an ontology that specifies:

- A class hierarchy
- Domains and ranges for all properties
- Declarations that properties are functional

For this paper we make a number of simplifying assumptions (that are not made in out full system):

- There are no property hierarchies. Property hierarchies complicate the matching.
- The class hierarchies are trees. Siblings in the class hierarchies form disjoint sets of individuals.
- The open world assumption: we do not assume that we are told everything about an individual. If we want to say that something is not true, we need to say that its value is absent.

[8] Our ontology specifies other information such as property hierarchies, and number ranges, that are outside of the scope of this paper.

– Properties are functional. Note that this also applies to the "type" property as an instance only has a single type (even if we don't know what it is).

We assume that we are given an instance or instances of the form:

$$\langle Object, Property, Value, Frequency\rangle$$

where $Frequency$ is either $present$ or $absent$ and $reference$ specifies the source of the information.

We assume that we are given a model or models of the form:

$$\langle Object, Property, Value, Frequency\rangle$$

where $Frequency$ is one of {*always, usually, sometimes, rarely, never*}.

In the models and instances, the $Values$ can be primitive values (such as strings or numbers or pair of numbers representing ranges), descriptions of other objects, or references to other objects.

For example, Figure 2 gives a model of a research meeting. As part of the ontology, we assume that the range of *starts_at* is a time. Thus we know that the type of t_1 must be a subclass of time. That is why it isn't specified here. Similarly for the location L_1.

Object	Property	Value	Frequency
M_1	*has_location*	L_1	usually
M_1	*starts_at*	T_1	always
M_1	*organized_by*	P_1	always
L_1	*type*	*Building*	always
L_1	*type*	*House*	rarely
T_1	*between*	[900, 1700]	always
T_1	*between*	[1200, 1259]	rarely
P_1	*type*	*Admin_staff*	usually
P_1	*type*	*Dept_head*	rarely
P_1	*type*	*Financial_officer*	never

Fig. 2. An example model of a research meeting

Note that the first line of this description allows for the fact that a meeting may not have a location (which is different from whether we know its location).

Under these assumptions, there is a canonical form for instances. Because the declarations are implicitly conjoined, you can assume there is exactly one "present" class for any functional property (including type) and a number of absent classes that are subclasses of the present class. This is because you can always assume that the top element is present, and if a class and a subclass are both present, you can remove the superclass as present and preserve the meaning. There can't be two classes that are both "present" if one is not a subclass of the other if the hierarchies are trees (as there are no elements in common between the classes).

Similarly we can assume that for a model, there is at most one *always*, at most one *usually*, that frequencies go down in the hierarchy, there are no children of *never* or cousins of *always*. The only cousins of a usually are *never*s. [Note that *sometimes* is used when we have a complete knowledge assumption; it will be ignored in this section.]

Figure 3 gives an algorithm to determine the score for an instance matching a model. We use the notation $I.present$ to be the position in the hierarchy of the value of the instance that is declared to be present. Similarly $M.always$ is the position in the hierarchy of the value declared to be always true in model M. Conditions involving $M.always$ are assumed to be false if nothing is declared to be always true in model M. Below and above refer to positions in the hierarchy (above is more general), and a node is below itself and is above itself.

One non-obvious aspect of this algorithm is when the model has an "always" above a "usually". In this case, if an instance has *present* below the the *always*, but a cousin of the *usually*, it gets just the usually-absent penalty, and no reward for the always-present. This is reasonable as, if the always was not there, this would be equivalent to having always at the top. If the instance has *present* above the *usually*, it has a reward that is linear in the depth of the $present$ instance, independent of the position of the *always*.

5 Matching Parts

In order to complete the algorithm, we need to consider the case where the value may be an object that has a complex description. In this case, when the model and the instance both have have a complex descriptions, we need to determine the match between these complex descriptions. If the instance has multiple instances of the property (and the property is not functional), we choose the best match (independently of the model frequency).

For example, consider the value of the *organized_by* property of M_1 in Figure 2. If there exists a person who fits the description of P_1, there should be a reward, and if there doesn't there should be a penalty.

Again it is constructive to consider a full probabilistic analysis of the probability of model M_1 given instance I_1:

$$\begin{aligned} P(M_1|I_1) &= P(M_1|(\exists P_1) \wedge I_1)P((\exists P_1)|I_1) \\ &\quad + P(M_1|(\neg\exists P_1) \wedge I_1)P((\neg\exists P_1)|I_1) \\ &= P(M_1|(\exists P_1))P((\exists P_1)|I_1) \\ &\quad + P(M_1|(\neg\exists P_1))P((\neg\exists P_1)|I_1) \end{aligned}$$

where $\exists P_1$ is shorthand for there exists a P_1 that matches the description of P_1 in Figure 2. The model gives the qualitative values $P(M_1|(\exists P_1))$. To compute the qualitative values of $P((\exists P_1)|I_1)$ we can find the best match of P_1 to the values of *organized_by* in I_1.

Taking the qualitative version of this formula, we replace multiplication by addition and the addition by minimum in the kappa calculus or maximum in our system. The works if we expect $(\exists P_1)$ to be true (the frequency of *organized_by* in M_1 is always or usually) and we find that we have positive support for $(\exists P_1)$.

procedure scoreMatch
Inputs:
    Model Description M
    Instance Description I
Returns:
    Score
begin
    if ($I.present$ is below a $M.never$)
        return *never-present* reward
    else if ($I.present$ is cousin of $I.always$)
        return *never-present* reward
    else if ($I.present$ is below a $M.rarely$)
        return *rarely-present* reward
    else if ($I.present$ is cousin of $I.usually$)
        return *rarely-present* reward
    else if ($I.absent$ is above $M.always$)
        return *always-absent* reward
    else if ($I.absent$ is above $M.usually$)
        return *usually-absent* reward
    else if ($I.present$ is below $M.usually$)
        return *usually-present* reward
    else if ($I.present$ is above $M.usually$)
        return $\alpha \times$ *usually-present* reward
            where α is depth of $I.present$ / depth of $M.usually$
    else if ($I.present$ is above $M.always$)
        return $\alpha \times$ *always-present* reward
            where α is depth of $I.present$ / depth of $M.always$
end

Fig. 3. Determining Reward from Instance and Model Descriptions

Unfortunately, the other 3 cases are not as straightforward as we cannot readily compute $kappa((\neg\exists P_1)|I_1)$ as if $(\exists P_1)$ has no surprise, then its negation has some surprise, but we don't know how much, and $(\exists P_1)$ has some surprise, then its negation has no surprise. We have chosen a simple scheme that gives intuitive results, as follows.

If the model specifies we are surprised that $(\exists P_1)$ (the frequency is "rarely") and we are not surprised that it is true in the instance (i.e., $(\exists P_1)|I_1$ is positive), we get the surprise of the rarely (the *rarely-present* reward).

If the qualitative probability of $(\exists P_1)$ kin I_1 is negative, we give the appropriate *always-absent*, *usually-absent*, …*never-absent* reward.

For example, if the instance had multiple organizers, we need to determine which one best fulfills the role specified in the model. In terms of the kappa calculus, the distribution over which instance fulfills the role becomes a minimization of surprised (maximization of scores). We do this by choosing the one with the highest score. Then we consider the model frequency and how surprised we are than an organizer of the appropriate type exists.

6 Conclusion

This paper has grown out of a project to build knowledge-based decision tools in various domains such as minerals exploration, geological hazards (landslides, earthquakes), land-use planning. We have needed to have qualitative reasoning and rich ontologies. We don't have the probabilistic knowledge or the utilities to do full decision theory, but have developed a system that uses a small but natural set of qualitative probabilities that can integrate with the ontologies being developed and with the sort of knowledge about instances and models that can be obtained. This paper outlined how we are handling cases where a functional relation has qualitative probabilistic constraints on what values it can take (some are more surprising than others). We have made some pragmatic choices that seem to work in practice, but there is much more theoretical and empirical work that needs to be carried out.

References

Darwiche, A. and Goldszmidt, M. [1994]. On the relation between kappa calculus and probabilistic reasoning, *UAI-94*, pp. 145–153.

Koller, D., Levy, A. and Pfeffer, A. [1997]. P-classic: A tractable probabilistic description logic, *AAAI-97*, Providence, RI, pp. 390–397.

McGuinness, D. L. and van Harmelen, F. [2004]. Owl web ontology language overview, *W3C Recommendation 10 February 2004*, W3C.
URL: *http://www.w3.org/TR/owl-features/*

Patel-Schneider, P. F., Hayes, P. and Horrocks, I. [2004]. Owl web ontology language: Semantics and abstract syntax, *W3C Recommendation 10 February 2004*, W3C.
URL: *http://www.w3.org/TR/owl-semantics/*

Pearl, J. [1989]. Probabilistic semantics for nonmonotonic reasoning: A survey, *in* R. J. Brachman, H. J. Levesque and R. Reiter (eds), *KR-89*, Toronto, pp. 505–516.

Smyth, C. and Poole, D. [2004]. Qualitative probabilistic matching with hierarchical descriptions, *Proc. Ninth International Conference on the Principles of Knowledge Representation and Reasoning (KR-04)*.
URL: *http://www.cs.ubc.ca/spider/poole/papers/KR04SmythC.pdf*

Spohn, W. [1988]. A general non-probabilistic theory of inductive reasoning, *UAI-88*, pp. 315–322.